# *ComLink User's Guide*

**by Will Price**

**Version 1.0b11 - May 21, 1996**

## Distribution

ComLink is distributed as shareware.  This means that the program is fully functional for 30 days after which you must register the software to continue using it.  There are two levels of registration: *Terminal* and *Host*.  The Host registration is for those who wish to use ComLink as a bulletin board system.  The Terminal registration is for all others.

Terminal registration is $40 US.  Registration entitles you to email announcements of interest to ComLink users, and access to selected beta versions from a private Internet FTP site.  Please include your email address with your registration.

To register for the Terminal version, please send $40 in US funds by check or money order payable to Will Price at:

Will Price
PO Box 641383
Los Angeles, CA 90064

Pricing for the Host registration has not yet been set because it has not been released, but you can expect that an upgrade will be available from Terminal registration.

## License

## Trademarks

CHAPTER 1

# A Long Road

## INTRODUCTION

Logging into an Apple II bulletin board system back when I was around 8 years old in 1980 started a long journey for me in the BBS world. I'm sure I had no clue that I would ever be writing a BBS for the Macintosh called Hermes eight years later. BBSs have had an enormous impact on my life not just as a user, but obviously as the creator of a BBS. I'm writing this document on the cusp of the release of the first major Macintosh BBS introduction in many years. The landscape has changed dramatically from what it was back in 1988 when I wrote Hermes. There are a lot more users, much faster computers, and methods of connection to BBSs that are orders of magnitude faster. There are also graphical user interfaces, and relatively notable corporations starting to realize the potential of the market. I've changed also from the 16 year old who wrote Hermes. I have found myself explaining the entire history of Mac BBSing to person after person recently. A lot of people in the BBS community today didn't even have computers when much of this occurred or have made false assumptions about these events, and I hope that I can do a good job of laying down the sequence that led us to the BBS world that we have today on the Macintosh.

## THE BEGINNING

The Macintosh was introduced January 24, 1984, but it was a long time before a BBS or even a modem became available. I got my first Macintosh 128K a few weeks after it came out. My Apple II had been taken away by my parents a year before because I had attempted to use it for things they didn't like [‘nuff said]. I was stressing for a computer. It was at least 6 months before modems were available and third parties had figured out how to use the modem cable and serial interface. Mind you, I was only 12 in 1984, so my memories are somewhat foggy of those times.

How can I forget, however, the introduction of **Mouse Exchange** BBS. Produced by a small company in Florida called Dreams of the Phoenix, the software was written by Michael Connick who later wrote the **Tabby** mailer software. My original disk says copyright 1984, but it basically wasn't in use until 1985 and never really became popular. It was a great first attempt. It had some simple menus and a basic monochrome text window with a status display. It was divided into multiple applications to perform various BBS maintenance functions. It was buggy, had very few features, and was just not a commercial quality application even for that time. In its defense, writing anything at that time for the Mac was a monumental achievement because it had to be written on a Lisa.

## WE'RE ALMOST GETTING REAL

The market began to taste what could be done with **Red Ryder Host**. Introduced at the same time around 1986 was **WWIV/Mac**. The wars that raged for a while about which was better grew quite heated.

Red Ryder Host(RRH) was written by Scott Watson who was somewhat of a Macintosh hero at the time for writing Red Ryder. Some people loved his software, others hated it. There weren’t many in between. RRH was divided into many applications, so you had to quit the application and run another one in order to perform sysop functions like user editing and menu editing. It’s main feature was its customizability. It had a completely hierarchical multi-level menu system. In other words, you could have a message section menu for Macintosh stuff with a message section menu under that for Games, a section under that for Adventure Games, and finally a section for every major game. It was very versatile in its menus. The setup applications all had horrendous interfaces including huge 50 pixel high buttons all over the place - a Scott Watson trademark. It was a pain for users to understand because every BBS was completely different. If one assumes all users want to call only one BBS or are willing to learn an entirely new command set and hierarchy for each BBS, perhaps it is a good idea. Judging from its comparative acceptance, most people didn’t assume that. It had no PC-ANSI color support, no Zmodem, single node, the list goes on. It was a lot better than Mouse Exchange though.

WWIV/Mac wasn’t exactly five mouse software either by today’s standards. It was also a single node system with no Zmodem and no color. The important improvement it brought was integration. Finally, sysops were able to run a BBS in one application and do all user editing and menu editing from within that application, or even by calling in from remote. The problem was that absolutely everything was done in the text interface of the BBS. The program had one window for users and one for status. In other words, there was basically no Macintosh interface.

WWIV/Mac was a port to the Macintosh of the popular IBM BBS program called WWIV by Wayne Bell. The port was done by Terry Teague. I believe it was a noble effort for something that he undoubtedly didn’t have much time for. At the time, I was young and nearly everyone I knew spent much of their time denigrating Mr. Teague on the boards for not updating WWIV/Mac more quickly. Apparently, Terry lost interest in WWIV/Mac or became too busy, and there have been no further updates since about 1989 at the release of 3.0. Given the violent flaming some of the users running WWIV/Mac were inclined towards, one might be able to understand perhaps some of the reason Terry lost interest.

The general public outcry for a new BBS at that time was deafening to those who wanted to listen. I heard them. I wanted it just as badly.

## HERMES

It was in that atmosphere that during my Junior year in High School in 1988, I began to think of creating Hermes. At the time, I was running a very small WWIV/Mac BBS and hated it. Every day Hermes was not done was another day I would loathe running WWIV/Mac. I must admit I was pretty worked up about the whole thing, and was completely naive in thinking that I could do it with so little programming experience. I had written one simple Control Panel called ScreenMaster that switched your startup screens randomly among a set of pictures. Needless to say it was not terribly complex.

Ignorance is bliss as they say. I merrily plodded along writing the BBS. By that time, WWIV on the PC had a major 4.0 upgrade. It had been completely rewritten, so it was really impossible for Terry Teague to update WWIV/Mac to the new feature set without a similar rewrite. It was also in the C programming language whereas WWIV/Mac (based on the 3.21 PC version) was written in Pascal. I chose to write Hermes in Pascal like any good Macintosh application back then, but at the same time I chose to write it such that it looked like the new version of WWIV on the PC with color ANSI graphics. Hermes was initially designed to be a single node system.

Actually, to say that Hermes was "designed" at all is really an overstatement. It was more like a continuous stream of patches. Being a professional Software Engineer now, I cringe at the kind of haphazard coding that went into Hermes. Usually, design comes before writing code. With Hermes, coding was equal to design. As I encountered a problem, I invented a patch to what I had done which attempted to solve it. Efficiency was never an issue. I always assumed everything would be fast enough. It was a truly naive design, but somehow it worked and was better than the other options out there.

After a few months of development, I decided that I really wanted to simply copy the "look and feel" of WWIV PC since that was what everyone liked and knew, obviously without using any of its source code since it was in a different language. At the time, Apple was suing Microsoft over look and feel issues and everyone felt very unsure about such things legally. I sent a personal letter by US Mail to Wayne Bell, the author of WWIV PC. In it, I requested his explicit permission to use the text online interface from WWIV. He sent back a kind note which stated that he had received many similar requests, but most of them really just wanted to steal his source code. He stated that if I sent him a printout of my data structures code to prove that I was not doing that, he would not be concerned. He also stated that he himself had lifted the interface for WWIV from an Apple II BBS, and that his "look and feel" was really in the public domain. I did send the source, and everything worked out well.

After only six months of development in December of 1989, the first BBSs running Hermes began to appear as I privately released the first version 0.75 to a few select beta testers with whom I'd been speaking for advice. It was a single node system, but it did have ANSI color graphics. It was the first BBS on the Mac to support that. Zterm was at the time the only terminal program which could be used for ANSI. About a month later, I released Hermes v0.82 wide to the public with support for up to 10 nodes. The method of doing multiple nodes was a complete hack done as a sort of experiment to see if it would work.

It was fundamentally the wrong way to do it, but there was no turning back.  Once all the code for users being online was written to that spec, it would have required a near complete rewrite to turn back.  I had no idea how wrong a decision I had made.  I'll explain later the exact problems with the Hermes method.

The race was on at that time as I became very excited at the influx of public acceptance that Hermes gained from the starving BBS users. The BBS market began to vibrate with new life.  Rumors of an update to Red Ryder Host began circulating.  A new BBS named **Mansion**, written in BASIC, was discussed by a few.  A BBS called **NovaLink**, written by Mark Weaver and Alex Hoppman, also had appeared on the scene.  Neither of those caught on.  Mansion did not catch on because it was very slow, single node, and had few features.

NovaLink was the opposite.  Users were not ready for it, and it had various features which were very confusing.  It suffered from the Red Ryder Host problem of endlessly hierarchical menu structures which turned every BBS that a user called into a new nightmare.  Under the hood, it was light years ahead of Hermes.  Few real users could see that because of its clumsy interface and it swiftly fell by the wayside as its authors moved on to other projects for the time being.  It was not quite dead yet however.

By March, rumors of an update to Red Ryder Host were burning up my mailbox.  It was the only viable competition to Hermes at the time, so I felt threatened.  They had a lot of money for development, so I imagined that they would be able to attack Hermes me somehow. Their great new feature was going to be Zmodem.  I decided I had to beat them to the punch.  It was an undertaking I was almost not up to at the time.  I enlisted the help of John Raymonds, who was willing to work for free in the interests of helping the BBS community.  He was a great Macintosh programmer responsible for such well-known Macintosh classics as **VisionLab** and **Dungeon of Doom**.  We worked together on an open external protocol interface.  I felt that what the Macintosh needed was an API which allowed any terminal program or host which wanted it to access a system level library of transfer protocols such as Xmodem and Zmodem.  I guess it was cool in concept, trying to implement the same thing which Apple's Communications Toolbox did two years later.  It didn't really work out.

John did a great job with what we had, but what we needed was for people experienced in protocol development to come along after us and write protocols to the spec we created.  Needless to say, just like with the Communications Toolbox, nothing good was ever written to the spec.  We were in a bind on Zmodem.  It was an incredibly undocumented protocol.  Unlike Xmodem for which a myriad of documents existed to describe the protocol internals, almost nothing except direct source code existed for Zmodem, which was a fairly new protocol at the time.  I emailed Dave Alverson of Zterm fame to ask if he would like to contribute his protocol (not the source) to our external transfer protocol interface.  He really seemed to take great offense at the very suggestion that he should do such a thing.  I still have no idea why he was so abrasive.  Hermes was a big reason most people I knew were using his program at all.  Due to his refusal, we were forced to seek other options.

**Termulator**, by Brad Quick, was our salvation.  I suspect few remember that program, but Brad graciously offered to donate the Zmodem source he had developed.  Fortunately, John Raymonds was able to work magic and implement it into a code module compatible with the Hermes external protocol interface.  It truly was magic because it had to be totally hacked to work with that interface.  It was then that I began to realize just how wrong some of the fundamental design decisions in Hermes were.  The Zmodem code was honestly not that stable, and putting it into our interface didn't make it any more stable.  Neither John nor I really knew how Zmodem should work and it wasn't even our code, so correcting bugs in it was really a mystery.

The end result, however, was that I was able to beat Red Ryder Host and make Hermes the first Macintosh BBS with Zmodem by about 1 month. The bugs in Hermes Zmodem are numerous and remain even today resulting from the rushed effort and fundamentally bad design at the core of Hermes.

My Senior year in High School was overtaken by Hermes development. I went to a Massachusetts boarding school all four years. We were not even allowed to have telephones! Thus, it was a bit odd for me to have a massive cellular antenna hanging out my window with a cell phone and an RJ-11 cellular jack interface hooked into a Telebit Trailblazer so that I could connect to the Hermes support BBS in Los Angeles at 1200 baud with MNP 5 error correction. It may sound lame today, but back in 1989 it was the height of technology and required major effort, installation, and expenditure. It was truly an adventure, but I was determined to run the support BBS. No other BBS software on the Macintosh had a support BBS run by its author, but I felt strongly that Hermes needed one. I also was constantly supporting users on the GEnie system which had graciously offered me a roundtable for Hermes there. I tried to hide the fact that it was so difficult for me to connect because I was afraid of revealing my age to the users.

During the first year, I spent probably an hour a day answering messages to support Hermes. Every day. Including weekends. Unfortunately, a combination of my ignorance of what it meant to give technical support with the average age of the Hermes users and the pretense of such age groups towards flaming resulted in a very bad technical support situation eventually. Technical support really is a business and takes tons of time. A key tenet for software authors should always be to have someone else do your technical support if possible. I tended to take everything personally. It was easy to offend me. Your average 15 year old user of Hermes usually wasn't in the mood to be patient when I didn't answer email for a few days. When you have 100 support messages coming in each day, a full-time

Senior load, and a very expensive 1200 baud cellular connection with a tendency to drop connection randomly, things are pretty hopeless on the support front.

During the Summer of 1990 after Hermes had reached a fairly stable point with Zmodem, multinode support, color ANSI, and quite a few users, I finally met Mark Weaver. Mark was the main author of NovaLink, at the time used by almost no one due to its complexity and bugs. The story of NovaLink's birth is pretty juicy, but I don't feel right telling it all. Basically, the authors were best friends, then they weren't, and so NovaLink fell apart. Alex Hoppman ended up getting the rights to NovaLink with Mark's consent. Mark saw the serious flaws in NovaLink. Mark is a really smart guy. He knew that what really needed to be done was a complete rewrite of NovaLink and he knew how to do it.

Mark was working for a Macintosh multiport serial card hardware company while taking time off from Brown University. I was about to start college near Los Angeles. We met up at the August MacWorld Expo in 1990 for the first time and spent the whole time cooped up in my hotel room planning our dream BBS software. Indeed, only eight months after Hermes' initial release to the public, I knew quite clearly that only a complete rewrite could really allow Hermes to be what I wanted it to be. Mark and I struck up a very good friendship, but many troubles lay ahead.

## THE NEW BBS

Mark and I were sidetracked by our respective lives, but we got back together over Christmas vacation at my parents' house in Los Angeles. We literally didn't leave the house. Every minute was spent designing the new program. I learned an amazing amount from Mark. We worked together with a synergy that was truly awesome, both of us were perfect for our roles in the project. We had little clear idea then what “the project” was, but at least this time around design was a major factor before coding. We began to code that Summer. Mark spent the entire Summer at my house. Truly amazing code was written during that time. We fully designed the dream BBS system.

Unfortunately, by the end of the Summer it became clear to me that our design was so amazing it might just never get done. I had to lay down the line. We had a lot of great code, and a lot of hard work done on design, but we were at least two or three years away from release of what we really wanted without hiring more programmers. I told Mark that I refused to work on the project further unless we somehow tried to make it a reality for users now. We had to release what we had done somehow. We needed financial support to keep ourselves going. Mark agreed with my reasoning quickly and we set to designing what would eventually be released as MacIntercomm. The technology involved in MacIntercomm truly goes well beyond what it appears as on the outside. It is a no-holds barred operating system on the inside with its own GUI, pre-emptive multitasking system, persistent virtual memory storage, and the list goes on. Overlaying this on top of the Macintosh OS allowed us to accomplish pre-emptive multitasking, our key advertised feature for MacIntercomm. Other near-finished technology we had waiting in the wings that was never even included with MacIntercomm was truly awesome. It was all intended as a temporary measure on the road towards the final BBS product. We both knew from personal experience with Hermes and NovaLink that a rotten foundation led to an unmaintainable product. Our foundation for MacIntercomm was built of solid gold.

Meanwhile, I was still updating Hermes now and then. Obviously, I had lost interest by that Summer in doing any real work on Hermes. There really wasn’t any point. Hermes is designed from the very core as what they call in Computer Science a “state machine”. This is one of its fundamental flaws aside from being written in Pascal. It is also where it differs most significantly from MacIntercomm’s technology. Let me make an analogy to reading a book. There are ten books. You have to read all of them as fast as possible and you can’t ignore any one for very long.  The Hermes method would read a random amount from a particular book, spend a long time reconfiguring its seating position, close the first book and put in a bookmark, and finally start reading from the next book, assuming one didn’t have to make a long trip to the bathroom inbetween (better known as the Finder in this analogy).  The MacIntercomm method would open all ten books at once, read a few words at a time from each of them and have almost no overhead switching from book to book, and it would take very short bathroom breaks.  MacIntercomm would also have the special feature of being able to read all the books while in the bathroom.  Its the same key difference between Macintosh System 7 and Apple’s forthcoming Copland operating system.  The technical terms for this are cooperative multitasking for the Hermes method, and pre-emptive multitasking for the MacIntercomm method.  Whether you write things one way or the other changes the requirements and design completely.  Some software companies are going to have a really hard time moving to Copland.  ComLink will be very easy to port.  There will be more code that needs to be removed than added.

## MACINTERCOMM

Implementing the operating system at the core of MacIntercomm was a true feat of accomplishment for us.  By the Summer of 1992 once Mark and I had finished college for the year, we had offices in Los Angeles under a new corporation Mercury Systems, Inc, a company line, 1 employee, packaging being designed, a manual being written, and we were preparing to release the product by the August Mac-World drop-dead date.  Money was flying into the project from me, and it either had to stop or support itself.  Money was the fatal flaw of the MacIntercomm project.  Mark needed money, our employees needed money, the packagers needed lots of money, and the expo booths cost way too much.  I was spending many thousands of dollars a month to start up the company. There was no business plan. I just assumed that there would be enough money to make it work.

We never truly finished even the subset product we had envisioned for MacIntercomm. We finished the manual, packaged it up, finished a 1.0 version on time, and told everyone that we would soon have the scripting language version completed. That was when the nightmare started. The scripting language had been entirely designed by Mark. I had no experience writing compilers at the time, so it was not clear to me just how ambitious a project it was. Mark's estimates were that it would take about a month for him to complete the scripting language. Our scripting language was really a complete development environment, similar to the way Java is a scripting language. It was not a small project.

The script compiler dragged on endlessly. College started back up for both of us. We had two employees at the office handling tech support and accounting, but everyone knew the only important thing was getting the product done with the script compiler. We were holding back on a full strength marketing campaign until we had the product we had designed and even documented. About a month into the semester, I laid down the line. Mark had made little progress. I told him that we needed to finish the product immediately as promised or cancel the project and shut down. He agreed to take the term off from Brown and come to stay at my parents' house again in Los Angeles so that he could work at the office.

The compiler was obviously harder than he had imagined. Meanwhile, money was still streaming in to the company to cover advertising, employees, offices, taxes, and countless other expenses. The supply of funds I had built up from Hermes just wasn't going to make it.

Mark thought it could be done by MacWorld SF in 1993. We geared up for that announcement. We made our booth look really cool with tall Roman columns, each with a built-in Macintosh. We then had two employees, one of whom was Steve Shannon, a real asset to the company. He shared our vision, and had a lot of energy. I credit him with keeping both Mark and I going during the hard times there.

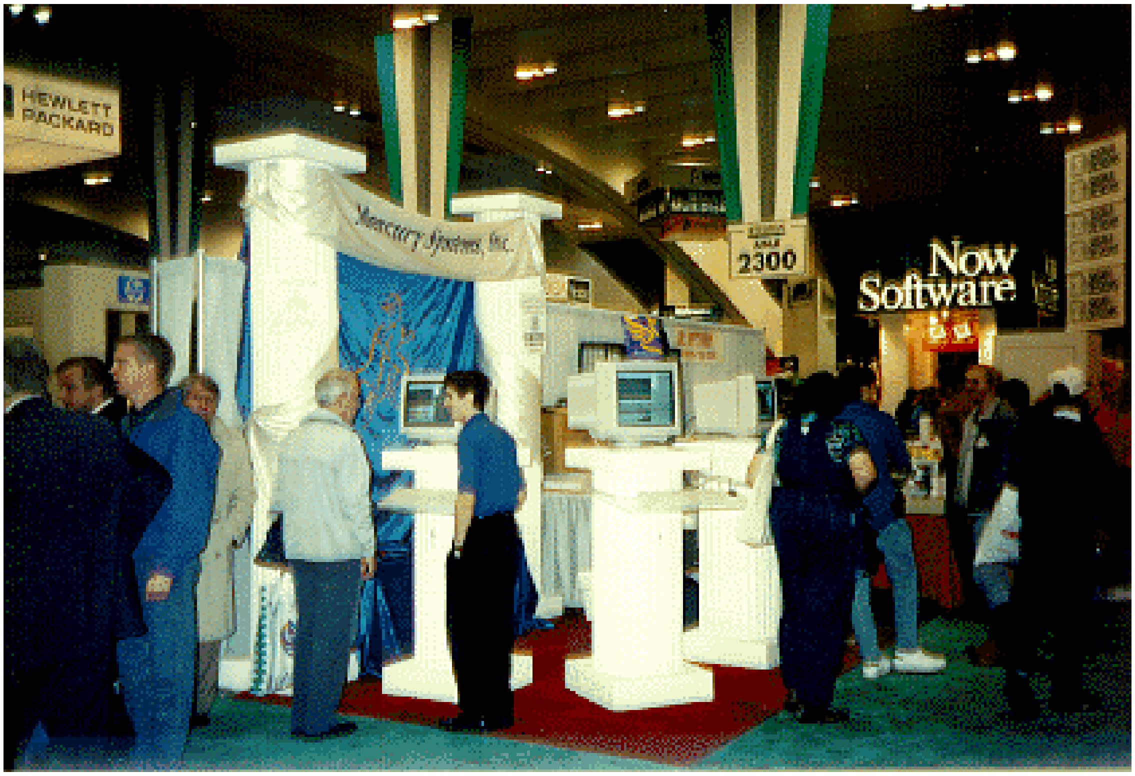


FIGURE 1. **Mercury Systems' booth at the 1993 MacWorld SF expo. Our computers were placed inside wooden Roman columns with clear plastic tablets connected to the columns to suspend the keyboard and mouse as if floating in mid-air.**

Even Steve couldn't help write the compiler obviously. We missed release at MacWorld SF. That was devastating. Everyone knew we had announced script compiler availability at the expo. My funds were depleted. I was really digging to support the company further.

My parents grew wary of my supporting it further.  Many friends advised me to bail out of the sinking ship.  There was little hope of completion for the script compiler by any reasonable date.  A month after MacWorld, I shut down the company forever.

I'm sure many can question my decision, but without money I was just digging my own grave.  There were innumerable back taxes and accounting errors that had built up which I ended up spending the next two years paying.  It was a hopeless cause without a major infusion of cash which was nowhere to be found.  It was really depressing.  Thousands of empty MacIntercomm boxes, manuals, disks, envelopes, stationery, packaging, desks, computers, our network, phone system: all of them sat in silence in our offices.

## EXIT

The breakdown of Mercury Systems removed any desire I had to write Macintosh software.  I thought I had done my best and it had failed miserably and therefore I should not try again.  Steve Shannon had other plans.  I wanted to get rid of everything.  He still wanted to see MacIntercomm happen.  He got a job at New World Computing, the producers of PC game software such as **Might and Magic**.  He was good friends with the management there, and soon convinced them to buy all the rights to MacIntercomm from me for a fixed sum plus royalties.  Mark had signed over all rights to me as an apology for his failure to complete the compiler.  Both of us wanted to exit the Macintosh software development community immediately.  I can't blame Mark for the problems at Mercury.  We lacked business experience and capital.  Without that, there was never any hope for success.

On March 31, 1993, I sold all the rights to New World.  I was also actively trying to get rid of Hermes.  As far as I was concerned then, Hermes and MacIntercomm were the same breed and it all left a bad taste in my mouth.  I told the Hermes beta team that I planned to sell the rights to Hermes. My beta testers since the beginning of Hermes' development were Dustin Bernard, Steve Shannon, Joe De Vita, Bob Taub, Charles Boozer, and Lloyd Woodall. I'm sure I've left out a few who came and went, but all of those stuck with it from the beginning.

Lloyd Woodall contacted me to express his interest in acquiring the rights to Hermes. I wanted nothing more than to get rid of Hermes as a constant burden every day reminding me about the Macintosh software industry. I sold it all to him for $20,000 and asked for no royalties. I had little confidence in what Lloyd would do with the product, so royalties were not important to me. I wanted only one continuing commitment from him. To quote the contract, "CC will list the name of 'W. Frank Price, III' in all future versions of the program, and the name will always be placed first and in at least as large a font as any other name appearing for credit. The name must appear int he About-Box, the Manual, and must be accessible by a remote user using into[sic] the Program over a modem. The name must always be identified as the Creator of the Program. The name must be identified as the 'Primary Author' for at least nine(9) months from the date of the signing of this Agreement."

Relations with Lloyd immediately fell apart. They were never very good while he was a beta tester, but at least he was fairly good at reporting bugs. He ran the Hermes BBS with the most nodes that I was aware of, a six line board in Seattle named PacMac, so I felt he was good to have as a tester.

I was just happy to be rid of it all. I sold all my Macs except for a PowerBook 180c and stopped any involvement with Macintosh software. I spent that Summer in Greece on a student program. That was where the healing started and I finally had enough free time to find myself. I'd been sealed up writing software for 6 years. It was high time to come up for air. By the time I returned to college for Senior year, I was a changed person. I was finally involved in a good personal relationship, and I was free to explore other areas. I graduated with a degree in Classical Languages in May 1994.

## REENTRY

During that year in school, I realized that I had overreacted to the failure of Mercury Systems. New World was doing very little with MacIntercomm. Lloyd was doing about the same with Hermes. Steve Shannon had left New World under complex circumstances, and there was no one there with the knowledge to keep it going. It was dead in the water. Lloyd had hired a programmer named Robert Rebbun to maintain Hermes. Lloyd had no programming experience, so it was always sort of a mystery to me how he planned to do anything with Hermes. Robert is a guy with an interesting history as it relates to Hermes.

Sometime in 1992, I sold a hard drive without wiping the data. The buyer of that hard drive undeleted everything off the drive and found the source code to Hermes v1.5. He then distributed it throughout the underground Macintosh community. Needless to say I was not pleased at the time. I thought nothing would come of it however. Six months later, a new BBS was released publically called **Proline**. I glanced at it and noticed that it looked an awful lot like Hermes. In fact, upon further inspection, most of the code was clearly lifted straight from Hermes v1.5.

I immediately contacted its “author” Robert Rebbun and demanded he remove all copies from BBSs and cease development. At first he refused to admit that he was using stolen Hermes source, but he soon gave in and all was forgiven. I thought that would be the last I heard about him, but Lloyd apparently thought that made him a great candidate to hire as the new programmer for Hermes.

By MacWorld Expo SF in January, 1994, I was excited about Macintosh again. I went to a party given by Steve Shannon with some of my old beta testers. It was there that I first decided to reenter the BBS and communications software market, but I was determined to correct the errors I had made previously.

## ComLink

Unfortunately, at that time I was in an especially bad position to be writing any communication related software for the Macintosh. I had sold most of the rights to both Hermes and MacIntercomm, and both contracts had a non-competition clause on the source code. It took a long time to work around those issues. I'd shoot myself before using any source code from Hermes. As shown in ComLink, I've programmed a WWIV-like interface ten times more efficient than Hermes. The source code from MacIntercomm was a big loss. I started writing ComLink under the assumption that I would have to develop everything clean-room without any rights to the MacIntercomm source. I started from scratch. There were some benefits to that approach. I got a chance to reanalyze some of the decisions which had made MacIntercomm's technology a little bit more complex than was required.

I felt that I had to try to get the rights to MacIntercomm back at least partially. Steve was no longer at New World, so it was very difficult. After nine months of negotiation, I worked out a deal with them to give me back the rights to the MacIntercomm source and eliminate the non-competition clause. Of course, I wasn't allowed to simply release MacIntercomm myself, but the source code I had slaved over with Mark was once again available for me to use freely. New World still has the rights to whatever they do with MacIntercomm. I have little hope that they will ever do anything.

So, for more than two years off and on I've been writing ComLink. It is a massive piece of code. We needed help on MacIntercomm desparately, and ComLink is an even more ambitious project. My basic tenet so far has been that if you want something done right, do it yourself. That's why it has taken so long. I spent the first nine months writing core sections of code from scratch and after getting the rights to MacIntercomm many of the parts of it that I wanted to use had to be adapted to the new technologies in ComLink.

I had the first betas of ComLink as a multiport terminal program ready for a select few testers in April of 1995. I was burned out. It had taken a lot out of me to take on the entire project myself, and I had to take a break. Fortunately, Philip Zimmermann came along and hired me to do **PGPfone**. That was very refreshing. It took almost six months out of ComLink development, but I learned a great deal and the project was quite worth it. I also took a month in January of 1995 to write **CryptDisk**. I've learned a great deal about cryptography, and I plan to implement some exciting security features in ComLink.

ComLink is nearing release. The terminal features have been in beta testing for over a year, and the BBS features are brand new. This is not what Mark and I originally envisioned for MacIntercomm — in fact, it's far from it. It is, however, a step in the right direction on a long path. The foundation for ComLink is very good. It is native on both 68K and PowerPC, and has been designed from the start as a multithreaded C++ object-oriented application that should port to Copland in record time. It is code that I enjoy working with, and therefore I believe it is the perfect platform to continue improving and through periodic updates eventually reach the vision of the dream BBS that Mark and I worked out in 1992. I also want ComLink to satisfy users' needs for a terminal application. The primary mistake I made with Hermes was bad design. That has most definitely been corrected.

Recently, Lloyd Woodall removed my credit from Hermes. I have decided not to demand that he put it back even though he is legally required to do so. I have no desire to be associated with that program any longer. It is a mess of code that should have been retired long ago. It certainly isn't worth porting to PowerPC native code, and it will never run efficiently under Copland without a rewrite from scratch.

It occurs to me that there are programs I have not mentioned such as **First Class**. It is not within the scope of this document to discuss whether First Class is a BBS or a LAN email product. I feel it is the latter and it seems that many sysops agree. Its market is not the BBS community. It makes for a really terrible BBS if the focus of the BBS is the transfer section. NovaLink Pro is still just NovaLink with some minor improvements. When Mark left that project, its innovation died. The primary mistake I made with MacIntercomm was lack of startup funding. ComLink will correct that by starting small. It will be distributed as shareware or keyware like Hermes was. Since there are really no continuing expenses involved in keyware, I can keep it going for a long time. The only problem is support. I will have to make sure that the Hermes situation is not recreated. The road ahead is bright for ComLink, and I'm eager to make it a reality.

CHAPTER 2

# Overview

This section is intended to give an overview of the features that ComLink offers. At the same time, it will attempt to explain how the various major concepts of ComLink fit together.

## *Product Highlights: Terminal*

ComLink is a full featured communications and host program for the Macintosh supporting **Serial**, **AppleTalk ADSP**, and **Telnet** connections. ComLink comes in two flavors, terminal and terminal/host. The keyware fee for the version that includes host features is higher than the low $40 fee for the terminal only version.

The 68K version of ComLink uses a technology called **pre-emptive multitasking** layered underneath the MacOS in a very safe manner to allow file transfers in ComLink to continue undisturbed regardless of what you may be doing in other applications. Other communications applications on the Macintosh (except MacIntercomm, see Chapter 1) generally let a transfer die or at least slow down if they don't get enough processor time.  This can occur from simple tasks such as booting applications, printing, playing games, or formatting disks. Any of these will generally cause other communications program to cause your file transfers to abort or lose speed.  ComLink is immune to that problem.  The PowerPC native version of ComLink does not have this feature.  In Apple's next system software release, it is widely expected that pre-emptive multitasking will be built-in to the operating system.  The engineering effort involved in making the PowerPC version do this now considering the impending release of Apple's new OS does not seem like a good investment of time.  The 68K version runs well on PowerPCs if you need the backgrounding capabilities, however, it is quite refreshing to watch the scrolling speed and other advantages of the PowerPC version.

ComLink supports the **VT52**, **VT102**, **VT220**, and 16 color **IBM PC-ANSI** screen emulations.

File transfers in ComLink are a strong point.  All file transfers are capable of the pre-emptive backgrounding feature using either **Xmodem**, **Ymodem**, **Ymodem-G**, **Zmodem**, **Kermit** with sliding windows and 9K long packets, and a proprietary bidirectional protocol called **Stripe**.

The **Stripe file transfer protocol** allows users to transfer files over a modem connection in both directions at the same time at speeds equal to or exceeding that of Zmodem.

**Scripting** in ComLink uses the **AppleScript** scripting language. Several samples demonstrate the powerful things that can be done with this language including a droplet to send files with a particular protocol making batch transfers as easy as drag and drop.

The terminal window in ComLink uses a technology called **Turbo-Draw** to place text on the screen at blazingly fast speeds.  Also, the scrollback buffer saves all information including color, blink state, double height/width, and all other emulation information.  The scroll-back buffer is also persistent meaning that the each time you run ComLink, everythign you've done before is still in your scrollback buffer up to a certain limit.  Months of communications sessions can be stored in your scrollback buffer and searched with the Find feature.

## *Product Highlights: Host*

This section left blank until the release of the host version of ComLink.

## *Concepts*

Two key terms should be defined at this point.

### PORT

A port in ComLink is represented by a terminal window.  A terminal window allows you to communicate over any of the communication methods that ComLink supports including Serial for modems, ADSP for AppleTalk networks, and Telnet for the Internet.  You can create new ports at any time and delete others.  A port is assocated with various preferences most notably the type of communications used by that port.  A Serial port will also have other information such as which Macintosh serial port to talk through (usually Modem or Printer).

## SERVICE

A service represents a system that you connect to with ComLink such as CompuServe, a local bulletin board system, or your shell account with your Internet provider. You should make a service for every service you wish to connect to with ComLink. Each service records information regarding many possible ways of contacting it. For instance, CompuServe can be accessed through the Internet by telnetting to “compuserve.com.” It can also be accessed by dialing the local access phone number. ComLink accepts both pieces of information and makes the decision on which to use by what connection type is specified for the port you choose to connect to CompuServe.

Once you understand the relationship between ports and services, setting up ComLink should be a snap.

CHAPTER 3

# Quickstart

This chapter is for users who want to jump right in and start using ComLink converting all their services and logon information from another program.

## Requirements

The system requirements for ComLink are not fully defined at this point. You will need to run at least System 7.0. For the PowerPC version of ComLink, you should run at least System 7.5 or an earlier system with the Threads Manager installed.

At this stage of beta, if ComLink does not run on your system or exhibits any odd behavior, please report it to the author at the address in the release notes so this section can be updated in the future.

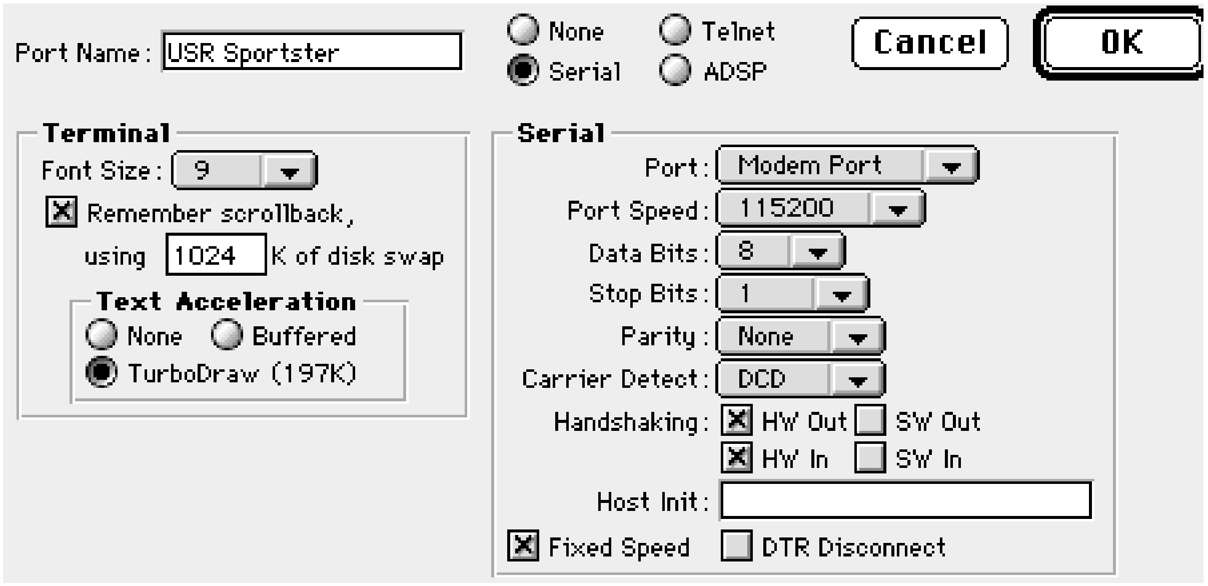
# *First Time Setup*

## SETTING UP A PORT

The first thing to do when running ComLink is setup your port preferences. The first time you run ComLink, it will create a default port for you which will try to connect to a modem on your modem port. If this fails, the port will hve no connection to anything. Clicking in the lower left corner of the window on the circle will bring up the port preferences. In this window, you should set the connection type you want ComLink to use.

We're going to setup ComLink for a single port connected to a US Robotics Sportster 288 connected to a PowerMac with the cable documented in Appendix A. Note that if you don't have the cable in Appendix A, certain feature of serial port connections with ComLink will not function properly. For instance, ComLink will not know whether you have disconnected from a service unless you tell it by selecting Disconnect from the menu. It is highly recommended that you obtain this cable as it will make your life quite enjoyable and bring new control to your terminal experience.

PowerMac serial ports can handle high baud rates, so making sure you have the SerialDMA 2.0 extension from Apple installed (included with System 7.5.3 and also available separately), set the options for hardware handshaking while making sure DTR Disconnect is disabled.



Port Name: USR Sportster

None [✓ Serial] Telnet ADSP

**Terminal**

Font Size: 9

[✓ Remember scrollback,] using 1024 K of disk swap

**Text Acceleration**

None Buffered [✓ TurboDraw (197K)]

**Serial**

Port: Modem Port

Port Speed: 115200

Data Bits: 8

Stop Bits: 1

Parity: None

Carrier Detect: DCD

Handshaking: [✓ HW Out] SW Out [✓ HW In] SW In

Host Init:

[✓ Fixed Speed] DTR Disconnect

FIGURE 2. **Port preferences window setup for a US Robotics Sportster 288 on a PowerMac with the modem cable in Appendix A**

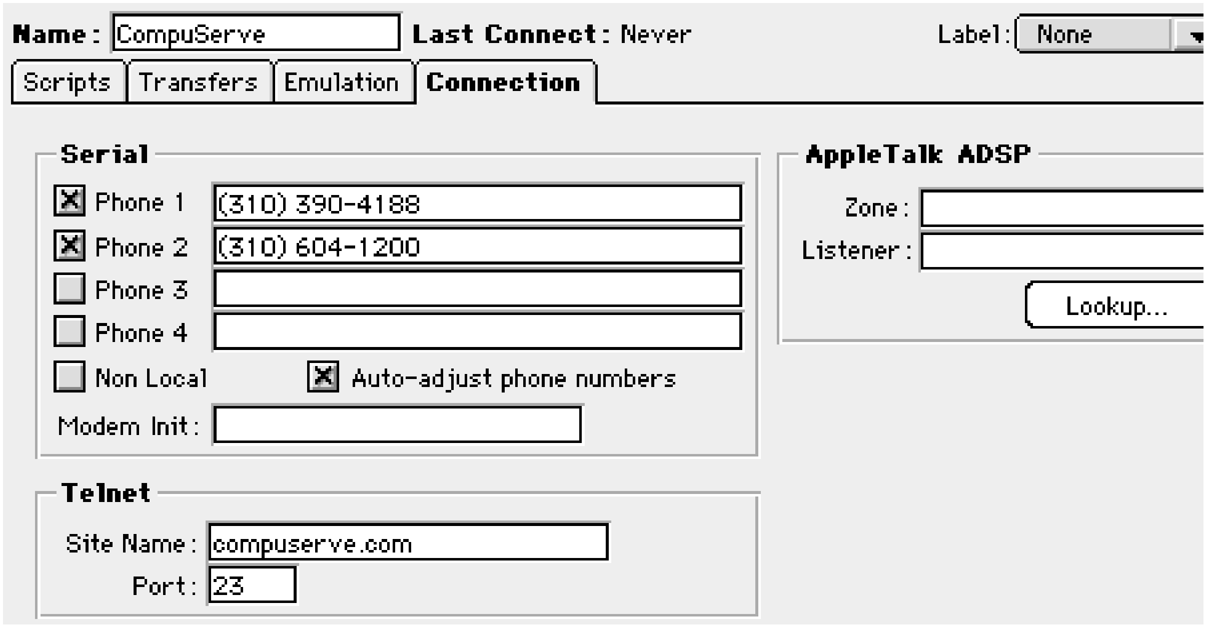
If you don’t have the cable from Appendix A, set the Carrier Detect option to None. ComLink will then fallback into a compatibility mode with older cables. A simple hardware handshaking cable such as the one that comes with your modem is highly unlikely to be the correct cable.

To use Telnet or ADSP, just select that from the connection types and you’re done setting up the port.

You can also set your scrollback buffer size here. ComLink uses a form of persistent virtual memory to store your scrollback buffer inbetween sessions if you enable this feature here. Up to 20 MB of scrollback buffer can be stored by ComLink, and it will always be accessible simply by using ComLink’s velocity enhanced continuous scrollbar or the Find feature. While this size will not be required in memory, eventually this size will be taken up on the hard drive with your System Folder on it. The data is stored in ComLink's preferences files in your Preferences folder.

## SETTING UP SERVICES

The next thing to do after configuring a port is to enter in all your services. Select New Service from the Service menu for each of the services you want to be able to connect to with ComLink.



Name: CompuServe
Last Connect: Never
Label: None

Serial
[✓] Phone 1 (310) 390-4188
[✓] Phone 2 (310) 604-1200
[ ] Phone 3
[ ] Phone 4
[ ] Non Local
[✓] Auto-adjust phone numbers
Modem Init:

AppleTalk ADSP
Zone:
Listener:

Telnet
Site Name: compuserve.com
Port: 23

**FIGURE 3. Service preferences window setup to call CompuServe with 2 different phone numbers over a modem or over Telnet**

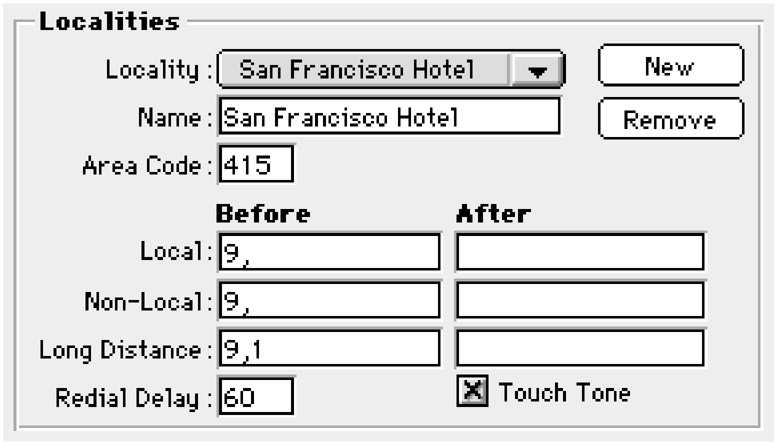
ComLink allows you to type in up to four phone numbers for each service. Many bulletin board systems and Internet providers have alternate phone number if the first is busy. When connecting to a particular service, ComLink will automatically try all four numbers in succession.  To activate a number, click the checkbox to the left of the number entry field.  If a certain number is disabled for a time, just uncheck that box.

Telnet addresses can be entered in the telnet area along with the port. Telnet uses port 23.  If you want to connect through some other protocol such as SMTP on port 25, you can change the port number here.

To switch between preference pages for the service, you can type Command-[1-4] as a shortcut with the numbers corresponding to the tabs, for instance 1 for Connection and 4 for Scripts.

Make sure to setup your other preference pages such as switching your emulation to either IBM PC-ANSI or VT220 as appropriate for the remote system.  For more information on the other preferences, please see the appropriate part of this manual.

The Auto adjust phone numbers checkbox uses ComLink’s automatic adjustment based on your area code to figure out what the correct number to call is.  For this feature to work, you must type in your area code in the main preferences dialog.  To disable this feature, uncheck this box.



| | | |
|---|---|---|
| Locality : | San Francisco Hotel | |
| Name : | San Francisco Hotel | |
| Area Code : | 415 | |
| | **Before** | **After** |
| Local : | 9, | |
| Non-Local : | 9, | |
| Long Distance : | 9,1 | |
| Redial Delay : | 60 | ☒ Touch Tone |

FIGURE 4. **General preferences window with the localities tab selected configured for a San Francisco hotel which requires dialing a 9 to dial out**

Once the localities window is configured, your services which use automatic adjustment of phone numbers will dial correctly. Whenever you change location, just set your new locality and ComLink will automatically remove or add back the area code, dial a 1, dial a 9, or whatever else you setup in the fields for that locality independent of the phone number for your services. This makes reconfiguration for travel very easy.

For quick access to your services, bring up the services window from the window menu. In general, a good idea is to keep your services window adjacent to your main port windows so you can easily double-click any service name to connect to it.

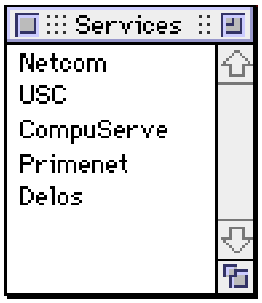


FIGURE 5. **Services window, double-click connects, option-click opens the Service preferences, and control-click switches to that service**

Each service is listed in the Services window. Single clicking on a service's name will select that service. A highlighted service is selected while a service name in bold means that is the "active" service. The active service is whatever service to which you last tried to connect. You can also make a service active by control-clicking on the service name.

To "**redial**" more than one service, use the standard Macintosh user interface method of selecting multiple items by command-clicking all the service names with which you would like to connect and then choosing "Connect Rotation" from the Port menu.

The terminal window in ComLink has a status area in the lower left corner which tells you the active service, and the status of a connection to that service. Clicking on the service name portion calls up the Service Preferences window. Clicking on the connection status icon on the left side brings up the Port Preferences window for the Port associated with the terminal.

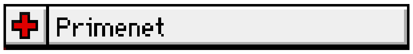


**FIGURE 6. Status area in disconnected mode**

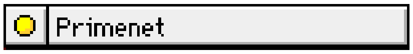


**FIGURE 7. Status area in connecting mode**

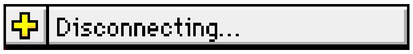


**FIGURE 8. Status area in disconnecting mode**

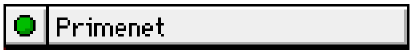


**FIGURE 9. Status area in connected mode**

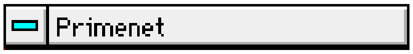


**FIGURE 10. Status area in listening mode**

CHAPTER 4

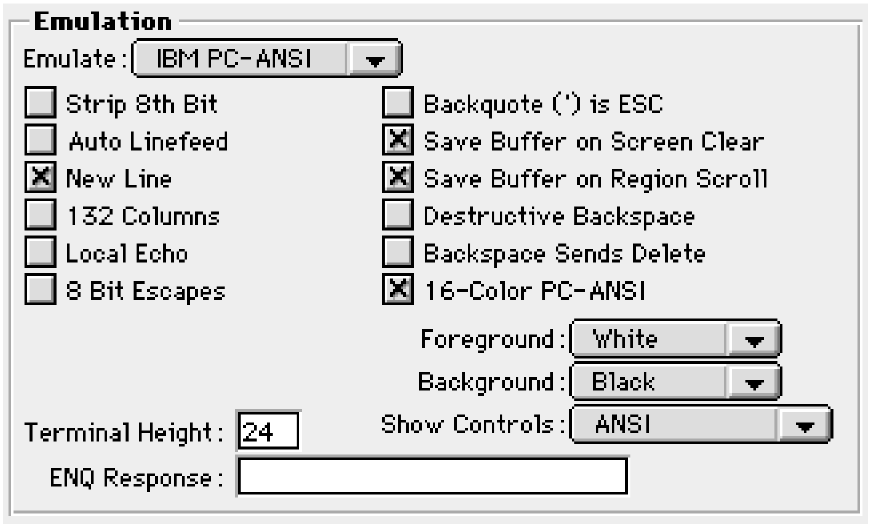
# *Services*

The concept of a Service in ComLink includes any systems you might connect to either by Serial modem connections, Telnet, or AppleTalk ADSP.

New Services are created by choosing **New Service…** from the Service menu. Services are deleted by clicking their name to select them from the Services window and then choosing **Remove Service** from the Service menu.

To edit the active Service, select **Service Preferences…** from the Service menu. You can also click the Service's name in the lower left corner of the terminal window in which that Service in active. Option-clicking the name of the service in the Service window also calls up the Service Preferences window.

## EMULATION OPTIONS



Emulation

Emulate : IBM PC-ANSI

☐ Strip 8th Bit
☐ Auto Linefeed
☒ New Line
☐ 132 Columns
☐ Local Echo
☐ 8 Bit Escapes
☐ Backquote (`) is ESC
☒ Save Buffer on Screen Clear
☒ Save Buffer on Region Scroll
☐ Destructive Backspace
☐ Backspace Sends Delete
☒ 16-Color PC-ANSI

Foreground : White
Background : Black
Show Controls : ANSI
Terminal Height : 24
ENQ Response :

FIGURE 11. **Emulation options pane from Service Preferences**

The **Emulate** menu allows you to select from ComLink’s terminal emulations. This setting is very important to achieve the proper appearance for your Services. Most bulletin board systems use PC-ANSI BBS, while most other systems will work well with the VT-220 setting.

**Strip 8th Bit** will strip the 8th bit in an 8 bit connection. This is useless if the Port is set to a 7 bit connection. This can be useful on systems like CompuServe which have a pseudo 7 bit connection. Setting the Port to 8 bits and checking this box can be a good idea in that case.

**Auto Linefeed** will automatically add a linefeed to incoming carriage returns. To begin a new line, both a return and a linefeed must be received. If you are communicating with a system that does not send a linefeed, checking this box will add one automatically if needed.

**Auto wrap** will automatically cause a full line of characters to wrap to a new line. Usually a full line (either 80 or 132 columns) with no returns will no longer be visible after the full line is printed, but with this feature everything will be seen.

**132 Columns** can be used with Services which allow DEC VT terminal widths of 132 columns. If checked, ComLink will enlarge the maximum width of the terminal to support this.

**Local Echo** will echo directly to the terminal window anything you type. Normally, ComLink will send everything you type only to the Service you are connected to or to your modem in Serial mode. This can be useful in two situations. The first is if you are directly connected to another individual's terminal program which does not echo back what you type. The second is if you are connected to a system such as GEnie which does not echo what you type.

**8 bit Escapes** sets ComLink to use the "CSI" 1 byte escape sequence for control codes rather than the 2 byte ESC-[ sequence normally used.

**Backquote (') is ESC** remaps your backquote key above the Tab key into the Escape key. This is useful on some Macintoshes which do not have an Escape key.

**Save Buffer on Screen Clear** places the contents of the terminal into the scrollback buffer whenever a screen clear occurs rather than simply clearing the data.

**Save Buffer on Region Scroll** places the contents of the terminal into the scrollback buffer whenever a region of the terminal window scrolls out of view. This can be useful in some cases, but in others such as editing with a UNIX program like “vi”, it can be bad since random lines will be saved in the buffer.

**Destructive Backspace** sends a backspace-space-backspace sequence rather than a simple backspace. This can be useful when directly connected to another Macintosh to allow you to erase characters you have typed on the remote terminal.

**Backspace Sends Delete** sends the Delete (127) ASCII character when you type the Delete key rather than the standard Backspace (8) character.

**16-Color PC-ANSI** sets ComLink to use 16 colors rather then the original PC-ANSI 8 colors. When the Intense option is set, the brighter colors will be used. This is how modern terminals work with PC-ANSI.

**Foreground** allows you to select the primary color of your foreground text.

**Background** allows you to select the primary background color of the terminal window.

**Show Controls** determines how ComLink will show control characters when that option is checked from the Port menu.

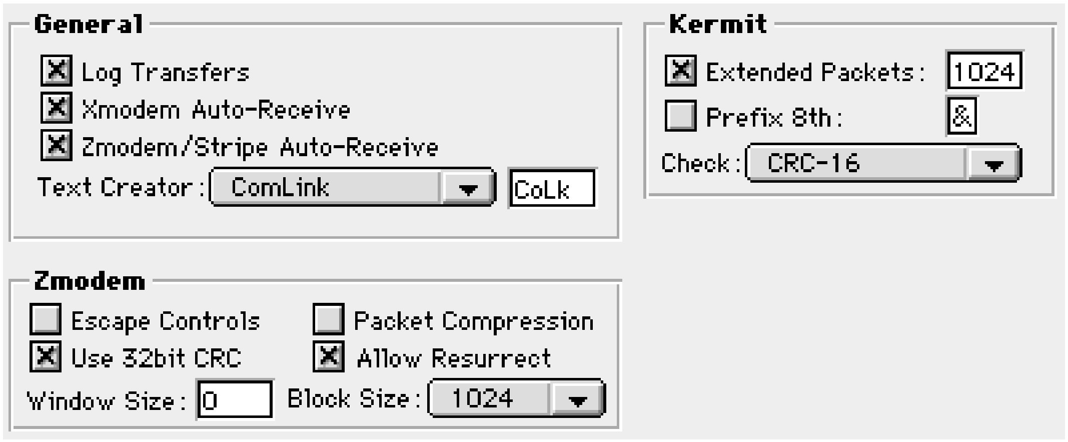
- **ANSI** will display control characters in their raw form as PC-ANSI control characters.
- **Highlighted** will display control characters as inverse text with the letter of the control character. For instance, Control-A would be displayed as an inversed text letter A.
- **Hexadecimal** will turn the ComLink terminal into a virtual line analyzer hex representation. It divides the window into an ASCII side and a hex side. This mode can be very useful for determining exactly what is being sent over the connection.

**Terminal Height** allows you to set the height of the terminal. This differs from simply the height of the window set when you resize it. This is the number of lines high used by the emulation. The standard default for almost every system is 24 lines. Using more can be useful on some UNIX systems which use text editors that allow a greater number of lines.

**ENQ Response** can be used to enter a text response to an ENQ query by a remote connection.

## TRANSFER OPTIONS

ComLink supports Xmodem, Ymodem, Ymodem-G, Zmodem, Kermit with the extended packets and sliding windows sometimes referred to as SuperKermit, and a proprietary bi-directional file transfer protocol called Stripe.



General
- [✓] Log Transfers
- [✓] Xmodem Auto-Receive
- [✓] Zmodem/Stripe Auto-Receive
- Text Creator: ComLink CoLk

Kermit
- [✓] Extended Packets: 1024
- [ ] Prefix 8th: &
- Check: CRC-16

Zmodem
- [ ] Escape Controls
- [ ] Packet Compression
- [✓] Use 32bit CRC
- [✓] Allow Resurrect
- Window Size: 0
- Block Size: 1024

FIGURE 12. **Transfer options pane from Service Preferences**

**Log Transfers** will add a line to a text file named “Transfer Log” with that statistics of each file transfer to or from this service. The “Transfer Log” file can be located either in the same folder as Com-Link or in the System Folder. The file looks similar to the following with another entry for each file transfer:

```
  Date      Time      cps    Elapsed   Size       Service         Filename
R:6/20/95   1:55 PM   2,548  00:00:32  81,606     LocalBBS       :MacTCP Watcher
R:7/21/95   10:32 PM  1,639  00:00:34  55,168     OtherBBS       :MacWEEK95072
```

**Xmodem Auto-Receive** activates the auto-receive MacBinary mode for Xmodem and Ymodem. This option is separated from the other auto-receive modes because it can sometimes be somewhat too easy to invoke unexpectedly.

**Zmodem/Stripe Auto-Receive** activates the auto-receive modes for the Zmodem and Stripe file transfer protocols, both of which require quite complex and unmistakeable sequences to initiate.

**Text Creator** allows you to specify the Macintosh file creator of any text files downloaded from this service.

**Escape Controls** tells the Zmodem file transfer protocol to escape all control characters int he data stream. This can be useful on some UNIX or VMS systems or over Telnet when the connection is not completely 8 bit.

**Packet Compression** enables a simple RLE compression for Zmo-dem. This can result in faster transfer times for text files. Many Zmodme implementations do not implement this feature.

**Use 32bit CRC** turns on the Zmodem 32bit CRC error checking rather than the alternate Zmodem 16bit CRC. This feature makes it more certain that the file packets do not contain errors.

**Allow Resurrect** permits Zmodem to search for existing ComLink partially received files in the active receive folder with the same name as the file beign received, and if found to resurrect the transfer.

**Window Size** allows you to tell Zmodem not to use its normal streaming protocol, and instead to use a windowing protocol which sends several packets up to the window size and waits for a positive acknowledgement. This can result in slower transfer times, but can be useful over connections which do not work with streaming due to flow control or other issues.

**Block Size** lets you set the maximum block size of a Zmodem packet. Generally, 1024 is a good size, but if your connection is particularly error prone, a lower block size may result in faster transfer times.

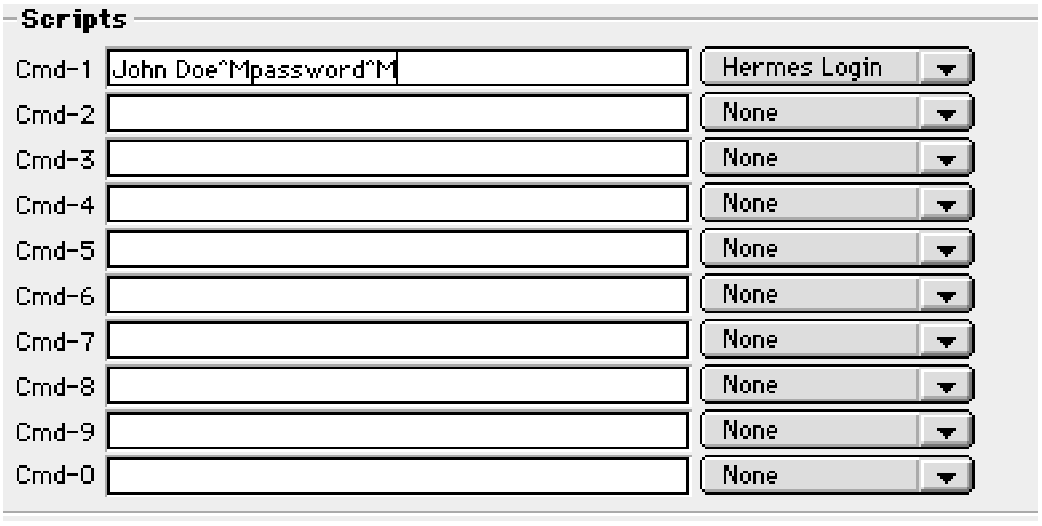
**Extended Packets** turns on the Kermit extended packets system. ComLink supports the maximum Kermit extended packets up to 9024 bytes in length. You can specify the maximum size in the field next to this checkbox.

**Prefix 8th** allows you to specify the character used to prefix escaped 8th bits in Kermit. The standard character to use is ‘&’, but occasionally some systems use different characters.

**Check** is a menu which allows you to specify the error control mechanism used for Kermit. Modern implementations of Kermit support a 6bit Checksum, a 12bit Checksum, and a 16bit CRC. The 16bit CRC will provide the best error detection.

## SCRIPTS

ComLink has an extensive scripting system with two levels. The top level is a simple macro system which allows you to associate a particular Command-# combination with a string. As shown, in this example typing Command-1 would execute the script called “Hermes Login”.  If no script was selected, the action would simply output the string in the text field.



| | | |
|---|---|---|
| Cmd-1 | John Doe^Mpassword^M | Hermes Login |
| Cmd-2 | | None |
| Cmd-3 | | None |
| Cmd-4 | | None |
| Cmd-5 | | None |
| Cmd-6 | | None |
| Cmd-7 | | None |
| Cmd-8 | | None |
| Cmd-9 | | None |
| Cmd-0 | | None |

FIGURE 13. **Script options from Service Preferences**

ComLink interprets the macro string according to the following rules:

Follow a ^ character with:

TABLE 1. **Translations of macro string codes**

| | |
|---|---|
| “ | “ |
| ^ | ^ |
| A-Z | Control A-Z |
| a-z | Control A-Z |
| [ | Escape |
| \ | File Separator(FS) |
| ] | Group Separator(GS) |

TABLE 1. **Translations of macro string codes**

| | |
|---|---|
| _ | Unit Separator(US) |
| ~ | Delete(ASCII 7F) |
| 0 | NULL |
| @ | NULL |
| < | Destructive Backspace (space, backspace, space) |
| / | CR-LF |
| $xx | Hardcoded hexadecimal value |

APPENDIX A

# *Modem Cables and Serial Ports*

Macintosh computers have traditionally been hobbled in the area of serial interfaces to modems.  The serial ports are so inferior to their PC counterparts that some have simply chosen to give up and declare that the Macintosh is unsuitable for use as a BBS.  To this day, not a single Macintosh is capable of supporting hardware ring detection. Carrier detect, DTR disconnection, and hardware handshaking remain issues that plague Macintosh users because the serial ports simply don’t provide the necessary interface to the modem hardware as IBM PC compatible serial ports do.

With the right cable, a PowerMac or AV Mac can reasonably solve all the problems stated above except hardware ring detection.  Only Apple can improve this situation, however, that is adequate for now.

Apple introduced in early 1996 a major update to its serial port driver software for PowerMac and AV Macs called SerialDMA 2.0, and it is built-in to System 7.5.3.  It is highly recommended that you make sure this is installed if it is applicable to your machine.

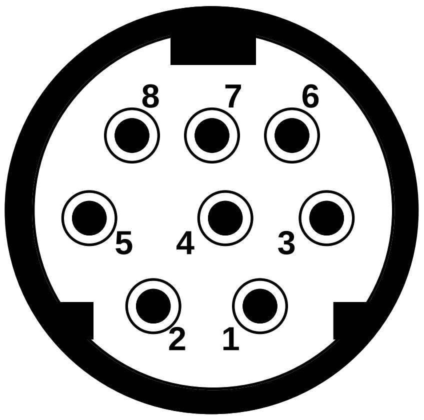


**FIGURE 1. Macintosh DIN-8 Serial Port Pinout**

The signals of the pins are as follows:

1. Output Hardware Handshaking (RTS/DTR)
2. Input Hardware Handshaking (CTS)
3. Data Transmit
4. Ground
5. Data Receive
6. Unused
7. General Purpose Input (GPI)
8. Unused

Pin 7 has only been connected on relatively recent Macintoshes. The collection of Macs which connect this pin is almost random, but generally a high-end or mid-range Macintosh does have the pin connected.

Having pin 7 connected is important. With it you can use ComLink to its fullest potential by making pin 7 Data Carrier Detect (DCD).

This allows ComLink to automatically determine whether a call is currently connected over the modem.  If ComLink can’t determine this, communications can sometimes be awkward.  This is a common problem most people have learned to accept about Macintoshes that those with PC compatibles have never needed to deal with.  ComLink takes full advantage of DCD if your setup allows it.

- Note that use of ComLink in BBS mode without properly configuring DCD will result in improper operation.  ComLink will be unable to detect a user unexpectedly disconnecting which will cause it to hang until you manually reset it.

Modem cables which have DCD hooked up on pin 7 can be obtained from select suppliers.  One such supplier is **MacDataflow** at (410)569-7636.  Any cable manufacturer should be able to custom manufacture the cable.  I designed the following cable in 1989 when I wrote Hermes BBS, and it has since become the de facto standard for bulletin board system operators on the Macintosh.

| DIN-8 | RS-232 |
|---|---|
| 1 | 4,20 |
| 2 | 5 |
| 3 | 2 |
| 4 | 7 |
| 5 | 3 |
| 6 | - |
| 7 | 8 |
| 8 | - |

**FIGURE 2. Pinouts for Hardware Handshaking/DCD cable**

Pin 6 is not connected, and pins 4 and 8 on the Macintosh side are jumpered together.  This cable works with all modems, and supports hardware handshaking, DTR disconnection, and Data Carrier Detect. Output hardware handshaking can’t be used with DTR disconnection because they use the same line as shown.  When using output hardware handshaking, make sure to disable DTR disconnection in your modem.  This is usually done by typing “AT&D0&W” in a terminal mode.

APPENDIX B

# Character Sets

The emulations in ComLink use unique character sets based on the standard ASCII. IBM PC-ANSI uses an 8bit code called Extended ASCII. The DEC VT emulations include a graphics character set and a more international character set. The following tables show the character sets in full.

To find the hexadecimal value of any character, take the row and then the column heading. For instance, for the PC-ANSI character set, the value of ‘A’ would be hexadecimal 41.

## EXTENDED ASCII PC-ANSI CHARACTER SET

| | 00 | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | A0 | B0 | C0 | D0 | E0 | F0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | NUL | DLE | SP | 0 | @ | P | ` | p | Ç | É | á | ░ | └ | ╨ | α | ≡ |
| 1 | SOH | XON | ! | 1 | A | Q | a | q | ü | æ | í | ▒ | ┴ | ╤ | ß | ± |
| 2 | STX | DC2 | " | 2 | B | R | b | r | é | Æ | ó | ▓ | ┬ | ╥ | Γ | ≥ |
| 3 | ETX | XOF | # | 3 | C | S | c | s | â | ô | ú | │ | ├ | ╙ | π | ≤ |
| 4 | EOT | DC4 | $ | 4 | D | T | d | t | ä | ö | ñ | ┤ | ─ | ╘ | Σ | ⌠ |
| 5 | ENQ | NAK | % | 5 | E | U | e | u | à | ò | Ñ | ╡ | ┼ | ╒ | σ | ⌡ |
| 6 | ACK | SYN | & | 6 | F | V | f | v | å | û | ª | ╢ | ╞ | ╓ | µ | ÷ |
| 7 | BEL | ETB | ' | 7 | G | W | g | w | ç | ù | º | ╖ | ╟ | ╫ | τ | ≈ |
| 8 | BS | CAN | ( | 8 | H | X | h | x | ê | ÿ | ¿ | ╕ | ╚ | ╪ | Φ | ° |
| 9 | HT | EM | ) | 9 | I | Y | i | y | ë | Ö | ⌐ | ╣ | ╔ | ┘ | Θ | ● |
| A | LF | SUB | * | : | J | Z | j | z | è | Ü | ¬ | ║ | ╩ | ┌ | Ω | ▪ |
| B | VT | ESC | + | ; | K | [ | k | { | ï | ¢ | ½ | ╗ | ╦ | █ | δ | √ |
| C | FF | FS | , | < | L | \ | l | \| | î | £ | ¼ | ╝ | ╠ | ▄ | ∞ | ⁿ |
| D | CR | GS | - | = | M | ] | m | } | ì | ¥ | ¡ | ╜ | ═ | ▌ | φ | ² |
| E | SO | RS | . | > | N | ^ | n | ~ | Ä | ₧ | « | ╛ | ╬ | ▐ | ε | ■ |
| F | SI | US | / | ? | O | _ | o | DEL | Å | ƒ | » | ┐ | ╧ | ▀ | ∩ | |

## DEC MULTINATIONAL CHARACTER SET

| | 00 | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | A0 | B0 | C0 | D0 | E0 | F0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | NUL | DLE | SP | 0 | @ | P | ` | p | | DCS | | ° | À | | à | |
| 1 | SOH | XON | ! | 1 | A | Q | a | q | | PU1 | ¡ | ± | Á | Ñ | á | ñ |
| 2 | STX | DC2 | " | 2 | B | R | b | r | | PU2 | ¢ | ² | Â | Ò | â | ò |
| 3 | ETX | XOF | # | 3 | C | S | c | s | | STS | £ | ³ | Ã | Ó | ã | ó |
| 4 | EOT | DC4 | $ | 4 | D | T | d | t | IND | CCH | | | Ä | Ô | ä | ô |
| 5 | ENQ | NAK | % | 5 | E | U | e | u | NEL | MW | ¥ | µ | Å | Õ | å | õ |
| 6 | ACK | SYN | & | 6 | F | V | f | v | SSA | SPA | | ¶ | Æ | Ö | æ | ö |
| 7 | BEL | ETB | ' | 7 | G | W | g | w | ESA | EPA | § | · | Ç | Œ | ç | œ |
| 8 | BS | CAN | ( | 8 | H | X | h | x | HTS | | ¤ | | È | Ø | è | ø |
| 9 | HT | EM | ) | 9 | I | Y | i | y | HTJ | | © | ¹ | É | Ù | é | ù |
| A | LF | SUB | * | : | J | Z | j | z | VTS | | ª | º | Ê | Ú | ê | ú |
| B | VT | ESC | + | ; | K | [ | k | { | PLD | CSI | « | » | Ë | Û | ë | û |
| C | FF | FS | , | < | L | \ | l | \| | PLU | ST | | ¼ | Ì | Ü | ì | ü |
| D | CR | GS | - | = | M | ] | m | } | RI | OSC | | ½ | Í | Ÿ | í | ÿ |
| E | SO | RS | . | > | N | ^ | n | ~ | SS2 | PM | | | Î | | î | |
| F | SI | US | / | ? | O | _ | o | DEL | SS3 | APC | | ¿ | Ï | ß | ï | □ |

DEC

SPECIAL GRAPHICS

CHARACTER SET

| | 00 | 10 | 20 | 30 | 40 | 50 | 60 | 70 |
|---|---|---|---|---|---|---|---|---|
| 0 | NUL | DLE | SP | 0 | @ | P | ◆ | ⎺ |
| 1 | SOH | XON | ! | 1 | A | Q | ▒ | ⎻ |
| 2 | STX | DC2 | " | 2 | B | R | ␉ | ─ |
| 3 | ETX | XOF | # | 3 | C | S | ␌ | ⎽ |
| 4 | EOT | DC4 | $ | 4 | D | T | ␍ | ├ |
| 5 | ENQ | NAK | % | 5 | E | U | ␊ | ┤ |
| 6 | ACK | SYN | & | 6 | F | V | ° | ┴ |
| 7 | BEL | ETB | ' | 7 | G | W | ± | ┬ |
| 8 | BS | CAN | ( | 8 | H | X | ␤ | │ |
| 9 | HT | EM | ) | 9 | I | Y | ␋ | ≤ |
| A | LF | SUB | * | : | J | Z | ┘ | ≥ |
| B | VT | ESC | + | ; | K | [ | ┐ | π |
| C | FF | FS | , | < | L | \ | ┌ | ≠ |
| D | CR | GS | - | = | M | ] | └ | £ |
| E | SO | RS | . | > | N | ^ | ┼ | · |
| F | SI | US | / | ? | O | | | DEL |